ANS Forth Internationalisation proposal

\\stephen\c\mpe\projects\international\i18n.propose.v6.doc
Revised 21 June 1999

Authors:
Stephen Pelc, MicroProcessor Engineering, sfp@mpeltd.demon.co.uk
Willem Botha, Construction Computer Software, willemb@ccssa.com
Nick Nelson, Micross Electronics, njn@micross.co.uk
Peter Knaggs, Bournemouth University, pjk@bcs.org.uk

Contributions from:
Greg Bailey, Athena Programming, greg@minerva.com

Contact:
Stephen Pelc
MicroProcessor Engineering
133 Hill Lane
Southampton SO15 5AF
England

Tel: +44 (0)2380 631441
Fax: +44 (0)2380 339691
Net: sfp@mpeltd.demon.co.uk
Web: http://www.mpeltd.demon.co.uk

Rationale

Forth Applications designed to run in many countries and languages cannot yet make enough assumptions about strings and character sets to be portable. The LOCALE word set is designed to provide words for portable internationalisation of application programs. The proposal does not attempt to cover text processing in general, but only to permit conversion of a limited set of application defined text for internationalisation.

In practice, many applications are not localised by the software developer, but by their agents in other countries. The LOCALE word set permits the software developer to provide tools that will produce text files that can be edited and converted to another language locally without dependency on computer language or operating system specific tools such as resource compilers and managers. At the same time, the proposed word set does not inhibit the use of sets of statically compiled strings for each language, it just does not define the mechanism.

The basis of the LOCALE word set is that all strings for internationalisation are compiled as LOCALE structures, and all access to the strings is through these structures. It appears that the following word set is adequate in the first place. The word set is designed to cope with character sets that are of different size to the native set.

The word set is split into a base and extension sets to indicate what factors need to be language sensitive. It is also likely that all LOCALE structures will need to be linked in case reindexing of hash tables or other internal structures is necessary.

The word **L”** is proposed for language sensitive strings, and behaves in a similar way to the ANS word **C”**, but returns a string identifier known as a locale string identifier (lsid) from which the required language string can be extracted. The reason for this is so that text information in the native development language is still available in the source, making source maintenance easier because the intention of the string is still available to the developer. In addition, the Forth compiler can be extended to produce a text file containing the native strings.

The number of items to be displayed which are, or may be, language sensitive is large. Not all applications may need to deal with all of them. In addition, many applications need to be able to perform text substitution, for example:

```
Your balance at <time> on <date> is <currency-value>.
```

We can provide for both these requirements by defining a text macro expansion facility. For example we can provide an initial string in the form:

```
Your balance at %time% on %date% is %currencyvalue%.
```

The % character is used as an escape, with %% returning a single % character. The text `time` `date` and `n` are text macro substitutions. The macros `time` and `date` insert the current time and date, and `currencyvalue` inserts the top item on the stack as a string in the current currency.

Terminology and assumptions

LOCALE

We use the word **locale** to mean the mixture of country, language, font, date/time formatting and so on in use when an application program runs.

Character sets

The language and character set encoding used by the Forth system at development time is referred to as the Development Character Set (**DCS**). The development character set is assumed never to change. It is furthermore assumed that character manipulation in the Forth system is defined in terms of the DCS, and that the action of character operations such as **`CMOVE`** is locked to the DCS.

The language and character set encoding used by any underlying operating system is referred to as the Operating Character Set (**OCS**). The OCS may or may not be the same as the DCS.

The language and character set encoding used at application run time is referred to as the Application Character Set (**ACS**). It is assumed that the largest character in an ACS fits in the native cell of the development Forth system. The only LOCALE word set use of individual characters is for setting macro escape characters (see later). The ACS may or may not be the same as the OCS.

The DCS is usually seven or eight bit ASCII in the majority of today's Forth systems, but we will see Unicode systems in the near future. The OCS is defined by the host machine, and is

defined by the user of the application. Thus, an application written in a Forth designed for ISO-Latin1 may be running on an O/S with a Chinese OCS, and a visitor may switch the application into yet another ACS, such as Russian. Such scenarios are rare within the US and Europe, but are common elsewhere in the world. Countries such as South Africa exist with 17 official languages, and some languages such as Portugese and English are spoken in many different countries.

LOCALE structures

We do not wish to constrain or influence implementation techniques in any way. A specific string for internationalisation needs to be referred to by a single parameter, which we call the "locale string identifier", or *lsid*. This is an opaque type, in other words the programmer should make no assumptions about what it means, except that different strings have different *lsids*. In many cases, an *lsid* may well be an address.

LOCALE strings

At application run time, locale strings need to be manipulated. Locale strings are described in terms of address units. For brevity, locale strings are also referred to as *lstrings*.

Country and language constants

There are a number of standardisation efforts for country and language codes. Since the objective of this document is to provide for source portability of applications, we do not need to mandate numeric or string values, but only to define language and country source names that can be used as Forth words.

Assuming that text processing is mostly affected by language selection, and that formatting is heavily influenced by both country and corporate standards, we suggest that country be defined by the ISO3166:1998 two letter country codes (Alpha-2). For this standard an algorithm has been defined to produce unique numeric codes for each country. A set of language codes also exists (ISO639:1998) also exists.

Octets and Bytes

Since the vast majority of character sets are defined in terms of 8 bit units commonly referred to as bytes or octets, it is likely that the implementation of any internationalisation code will require the presence of byte/octet access words, regardless of the underlying DCS character size.

The presence and definition of an octet/byte access mechanism is outside the scope of this proposal.

Macro substitution

The following three words are defined to handle macro substitution of text, and output of application data such as date, time, currency and so on. The normative definitions appear later in this proposal.

Proposed macro words (non-normative)

SUBSTITUTE \ i*x addr1 len1 addr2 len2 -- addr2 len3 j*y
Perform macro substitution on the *lstring* at addr1/len1 placing the result at *lstring* addr2/len2, returning addr2 and len3, the length of the resulting string. Ambiguous conditions occur if the resulting string will not fit into addr2/len2 or macro text cannot be found. An ambiguous condition occurs if addr2 is the same as addr1.

When a macro name delimited by escape characters (see **SET-ESCAPE**) is encountered by **SUBSTITUTE**, the following action occurs:

1) If the name is a valid macro name, a locale and implementation dependent action occurs
2) If the name is null, a single escape character is substituted
3) In all other cases an ambiguous condition exists

SET-MACRO \ addr len(au) c-addr u –
*** should this be an *lsid* ***
Define the localised string *addr/len(au)* as the text to substitute for the macro of the name (in the development character set) *c-addr/u*. If the macro does not exist it is created.

SET-ESCAPE \ locale-char --
Set the macro escape character to be the localised character *locale-char*. By default it is the ASCII % character if it is available in the application character set.

Rationale

The number of items which are, or may be, language sensitive is large. Not all applications may need to deal with all of them. In addition, many applications need to be able to perform text substitution, for example:

```
Your balance at <time> on <date> is <currency-value>.
```

Both these requirements are accommodated by defining a text macro expansion facility. We can provide the initial string in the form:

```
Your balance at %time% on %date% is %currencyvalue%.
```

The % character is used as an escape, with %% returning a single % character. The text `time` `date` and `n` are text macro substitutions. The macros `time` and `date` insert the current time and date, and `currencyvalue` inserts the top item on the stack as a string in the current currency.

The only word **required** to achieve this is **SUBSTITUTE**, although the other two add considerable flexibility.

Implementation of **SUBSTITUTE** may be considered as being equivalent to a wordlist which is searched. If the text is found, the word in the wordlist is executed to return a string which is substituted for the macro name. Such words can be deferred or multiple wordlists can be used. The implementation techniques required are similar to those many people have used to implement **ENVIRONMENT?**. The range of functions required can be handled by standardising macro names.

Advantages of this approach are that it only adds one word to the standard, and it does not mandate any implementation techniques. It is also extensible by the user without affecting the standard. The downside to the approach is that it requires an escape character.

It is recommended that the macro names and parameter sequences from ISO WD2 15435 (or later) be used where appropriate.

Note, however, that we cannot normatively define macros in terms of wordlists, as these require the presence of the development character set (DCS), which may not be a subset of the application character set (ACS).

The optional LOCALE word set

Environmental queries

Append the table below to table xxx

String value	Data type	Constant?	Meaning
LOCALE	Flag	No	LOCALE word set present
LOCALE-EXT	flag	No	LOCALE extension word set present

Additional documentation requirements

Ambiguous conditions

- use of an invalid locale string identifier (*lsid*)
- a locale string is too big for a destination buffer
- a macro name is not found by **SUBSTITUTE**
- an invalid locale string identifier (*lsid*) is used
- in-place macro subsitution

LOCALE words

SET-LANGUAGE \ lang – ior ; lang is a language code
*** change lang and country to an existing type (U) ***
Sets the current language for the LOCALE system. The *ior* is returned false if the operation succeeds, otherwise it returns a non-zero implementation-dependent *ior*. If the operation does not succeed, the current language remains unchanged.

GET-LANGUAGE \ -- lang
Returns the language code last set by **SET-LANGUAGE**. The default language is implementation defined.

SET-COUNTRY \ country – ior ; country is a country code
Sets the current country for the LOCALE system. The *ior* is returned false if the operation succeeds, otherwise it returns a non-zero implementation-dependent *ior*. If the operation does not succeed, the current country remains unchanged.

GET-COUNTRY \ -- country
Returns the country code last set by **COUNTRY**. The default language is implementation defined.

L” \ -- ; -- lsid ; **L” <native text>”**
Interpetation:
The interpretation semantics for this word are undefined.

Compilation: \ "ccc<quote>" --
Parse *ccc* delimited by a " (double-quote) and append the run-time semantics given below to the current definition.

Runtime: \ -- lsid
Return *lsid*, an identifier for a locale string. Other words use *lsid* to extract language specific information.

LOCALE@ \ lsid -- addr len(au)
Return the address and length in address units of the string (in the current language) that corresponds to the native string identified by *lsid*. The format of the string at *addr* is implementation dependent.
*** move next sentence to rationale and discuss alignment ***
The length of the string is returned in address units so that it may be copied by **MOVE** without knowledge of the character set width.

SUBSTITUTE \ i*x addr1 len1 addr2 len2 – j*y addr2 len3
Perform macro substitution on the *lstring* at addr1/len1 placing the result at *lstring* addr2/len2, returning addr2 and len3, the length of the resulting string. Ambiguous conditions occur if the resulting string will not fit into addr2/len2, or macro text cannot be found, or if the *lstring* at addr2/len2 overlaps the *lstring* at addr1/len1.

*** Discuss I*x and j*x *** - see EXECUTE

When a macro name delimited by escape characters (see **SET-ESCAPE**) is encountered by **SUBSTITUTE**, the following action occurs:
1) If the name is a valid macro name, a locale and implementation dependent action occurs
2) If the name is null, a single escape character is substituted
3) In all other cases an ambiguous condition exists

SET-MACRO \ addr len c-addr u --
Define the localised string *addr/len* in address units as the text to substitute for the macro of the name (in the development character set) *c-addr/u*. If the macro does not exist it is created.

SET-ESCAPE \ locale-char --
Set the macro escape character to be the localised character *locale-char*. By default it is the ASCII % character if it is available in the application character set.

LOCALE extension words

These words are provided here to give portability of implementation techniques. They are building blocks for a practical implementation.
*** more words and rationale ***

LOCALE-INDEX \ lsid --
Updates the internal data structure. Useful if structures are added and changes to internal structures are required. *** more words ***

LOCALE-LINK \ lsid1 -- lsid2
Given the address of one LOCALE structure, returns the address of the next. *** more words ***

LOCALE-TYPE \ addr len --
Displays the LOCALE string whose address and length in address units are given.

NATIVE@ \ lsid -- c-addr len
Given a LOCALE structure, returns the address and length of the corresponding DCS native string that was compiled by **L"**.

Compliance and labelling

Change history

21 June 1999
- Wordsmithed at ANS meeting

20 June 1999
- Tightened up some wording
- Added references to more standards.

14 June 1999
- Added an ambiguous condition to **`SUBSTITUTE`**.
- Changed **`COUNTRY`** and **`LANGUAGE`** to **`SET-COUNTRY`** and **`SET-LANGUAGE`** returning an *ior*.

30 May 1999
- Derived from parallel discussion document